

FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFFFFFFFF.FFF	111	111		
FFFFFFFFFFFFFFFF	111	111	XXX	
FFFFFFFFFFFFFFFF	111	111	XXX	
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111		
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111111111	111111111	XXX	XXX
FFF	111111111	111111111	XXX	XXX
FFF	111111111	111111111	XXX	XXX



```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```



```
1 0001 0 MODULE MAKPTR (
2 0002 0 LANGUAGE (BLISS32),
3 0003 0 IDENT = 'V04-000'
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
12 0012 1 * ALL RIGHTS RESERVED. *
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
19 0019 1 * TRANSFERRED. *
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
23 0023 1 * CORPORATION. *
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: F11ACP Structure Level 2
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 This routine constructs retrieval pointer in the file header map area
38 0038 1 for the indicated blocks.
39 0039 1
40 0040 1 ENVIRONMENT:
41 0041 1
42 0042 1 STARLET operating system, including privileged system services
43 0043 1 and internal exec routines.
44 0044 1
45 0045 1 --
46 0046 1
47 0047 1
48 0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 14-Dec-1977 17:13
49 0049 1
50 0050 1 MODIFIED BY:
51 0051 1
52 0052 1 B0103 ACG0122 Andrew C. Goldstein, 17-Jan-1980 15:54
53 0053 1 Get bug check codes up to date
54 0054 1
55 0055 1 B0102 ACG0096 Andrew C. Goldstein, 18-Dec-1979 18:59
56 0056 1 Add zero count bug trap
57 0057 1
```



```

58 0058 1 | B0101 ACG0008 Andrew C. Goldstein, 28-Dec-1978 19:27
59 0059 1 | Add placement control support
60 0060 1 |
61 0061 1 | B0100 ACG00001 Andrew C. Goldstein, 10-Oct-1978 20:00
62 0062 1 | Previous revision history moved to [F11B.SRC]F11B.REV
63 0063 1 | **
64 0064 1 |
65 0065 1 |
66 0066 1 | LIBRARY 'SYSSLIBRARY:LIB.L32';
67 0067 1 | REQUIRE 'SRC$FCPDEF.B32';
68 1058 1 |
69 1059 1 | LINKAGE
70 1060 1 | L_MAKE_POINTER = CALL :
71 1061 1 | GLOBAL (MAP_POINTER = 9),
72 1062 1 |
73 1063 1 | L_CHECK_POINTER = CALL :
74 1064 1 | GLOBAL (HEADER = 8);
75 1065 1 |
76 1066 1 | FORWARD ROUTINE
77 1067 1 | MAKE_POINTER : L_MAKE_POINTER, ! build map pointer
78 1068 1 | CHECK_POINTER : L_CHECK_POINTER; ! check if pointer will fit

```

```

80 1069 1 GLOBAL ROUTINE MAKE_POINTER (COUNT, LBN, FILE_HEADER, PLACEMENT_CODE) : L_MAKE_POINTER =
81 1070 1
82 1071 1 ++
83 1072 1
84 1073 1 FUNCTIONAL DESCRIPTION:
85 1074 1
86 1075 1     This routine constructs retrieval pointer in the file header map area
87 1076 1     for the indicated blocks.
88 1077 1
89 1078 1
90 1079 1 CALLING SEQUENCE:
91 1080 1     MAKE_POINTER (ARG1, ARG2, ARG3, ARG4)
92 1081 1
93 1082 1 INPUT PARAMETERS:
94 1083 1     ARG1: block count
95 1084 1     ARG2: start LBN
96 1085 1     ARG3: address of file header
97 1086 1     ARG4: if present, placement data to record or 0
98 1087 1
99 1088 1 IMPLICIT INPUTS:
100 1089 1     R9 = address to receive pointer
101 1090 1
102 1091 1 OUTPUT PARAMETERS:
103 1092 1     NONE
104 1093 1
105 1094 1 IMPLICIT OUTPUTS:
106 1095 1     R9 = address past pointer
107 1096 1
108 1097 1 ROUTINE VALUE:
109 1098 1     1 if OK
110 1099 1     0 if pointer won't fit
111 1100 1
112 1101 1 SIDE EFFECTS:
113 1102 1     pointer added to header map area
114 1103 1
115 1104 1 --
116 1105 1
117 1106 2 BEGIN
118 1107 2
119 1108 2 EXTERNAL REGISTER
120 1109 2     MAP_POINTER      = 9 : REF BBLOCK; ! pointer to map area
121 1110 2
122 1111 2 MAP
123 1112 2     FILE_HEADER      : REF BBLOCK; ! buffer containing file header
124 1113 2
125 1114 2 GLOBAL REGISTER
126 1115 2     HEADER            = 8 : REF BBLOCK; ! local pointer to file header
127 1116 2
128 1117 2 LOCAL
129 1118 2     PLACEMENT         ! copy of placement data
130 1119 2     POINTER_SIZE;     ! size needed for map pointer
131 1120 2
132 1121 2
133 1122 2 ! Check the specified count; it must not be zero.
134 1123 2 !
135 1124 2
136 1125 2 IF .COUNT EQL 0
```



```
137 1126 2 THEN BUG_CHECK (MAPCNTZER, FATAL, 'Attempted to generate zero length map pointer');
138 1127 2 ! See if placement is specified.
139 1128 2 !
140 1129 2
141 1130 2 PLACEMENT = 0;
142 1131 2 POINTER_SIZE = 2;
143 1132 2 IF ACTUALCOUNT GEQU 4
144 1133 2 THEN
145 1134 2 BEGIN
146 1135 2 PLACEMENT = .PLACEMENT_CODE;
147 1136 2 IF .PLACEMENT NEQ 0
148 1137 2 THEN POINTER_SIZE = .POINTER_SIZE + 1;
149 1138 2 END;
150 1139 2
151 1140 2 ! Get the address of the file header for the check routine.
152 1141 2 ! Then determine the format of the pointer and build it.
153 1142 2 !
154 1143 2
155 1144 2 HEADER = .FILE_HEADER;
156 1145 2
157 1146 2 IF .COUNT LEQU 256 AND .LBN LSSU 1^22
158 1147 2 THEN
159 1148 2 BEGIN
160 1149 2 IF NOT CHECK_POINTER (.POINTER_SIZE) THEN RETURN 0;
161 1150 2 IF .PLACEMENT NEQ 0
162 1151 2 THEN
163 1152 2 BEGIN
164 1153 2 (.MAP_POINTER)<0,16> = .PLACEMENT;
165 1154 2 MAP_POINTER = .MAP_POINTER + 2;
166 1155 2 END;
167 1156 2
168 1157 2 MAP_POINTER[FM2$V_FORMAT] = FM2$C_FORMAT1;
169 1158 2 MAP_POINTER[FM2$B_COUNT1] = .COUNT - 1;
170 1159 2 MAP_POINTER[FM2$V_HIGHLBN] = .LBN<16,6>;
171 1160 2 MAP_POINTER[FM2$W_LOWLBN] = .LBN<0,16>;
172 1161 2 MAP_POINTER = .MAP_POINTER + 4;
173 1162 2 END
174 1163 2
175 1164 2 ELSE
176 1165 2 BEGIN
177 1166 2 POINTER_SIZE = .POINTER_SIZE + 1;
178 1167 2 IF .COUNT LEQU 16384
179 1168 2 THEN
180 1169 2 BEGIN
181 1170 2 IF NOT CHECK_POINTER (.POINTER_SIZE) THEN RETURN 0;
182 1171 2 IF .PLACEMENT NEQ 0
183 1172 2 THEN
184 1173 2 BEGIN
185 1174 2 (.MAP_POINTER)<0,16> = .PLACEMENT;
186 1175 2 MAP_POINTER = .MAP_POINTER + 2;
187 1176 2 END;
188 1177 2
189 1178 2 MAP_POINTER[FM2$V_FORMAT] = FM2$C_FORMAT2;
190 1179 2 MAP_POINTER[FM2$V_COUNT2] = .COUNT - 1;
191 1180 2 MAP_POINTER[FM2$L_LBN2] = .LBN;
192 1181 2 MAP_POINTER = .MAP_POINTER + 6;
193 1182 2 END
```

```
194 1183 4  
195 1184 3 ELSE  
196 1185 4 BEGIN  
197 1186 4 POINTER_SIZE = .POINTER_SIZE + 1;  
198 1187 4 IF .COUNT LEQU 1^30  
199 1188 4 THEN  
200 1189 5 BEGIN  
201 1190 5 IF NOT CHECK_POINTER (.POINTER_SIZE) THEN RETURN 0;  
202 1191 5 IF .PLACEMENT NEQ 0  
203 1192 5 THEN  
204 1193 6 BEGIN  
205 1194 6 (.MAP_POINTER)<0,16> = .PLACEMENT;  
206 1195 6 MAP_POINTER = .MAP_POINTER + 2;  
207 1196 5 END;  
208 1197 5  
209 1198 5 .MAP_POINTER = ROT (.COUNT-1, 16);  
210 1199 5 MAP_POINTER[FM2$V_FORMAT] = FM2$C_FORMAT3;  
211 1200 5 MAP_POINTER[FM2$L_LBN3] = .LBN;  
212 1201 5 MAP_POINTER = .MAP_POINTER + 8;  
213 1202 5 END  
214 1203 5  
215 1204 4 ELSE BUG_CHECK (PTRCNT, FATAL, 'ACP block count exceeds retrieval pointer size');  
216 1205 3 END;  
217 1206 3 END;  
218 1207 2  
219 1208 2 RETURN 1;  
220 1209 2  
221 1210 1 END;
```

! end of routine MAKE\_POINTER

				.TITLE	MAKPTR	
				.IDENT	\V04-000\	
				.EXTRN	BUG\$_MAPCNTZER, BUG\$_PTRCNT	
				.PSECT	\$CODE\$,NOWRT,2	
				.ENTRY	MAKE_POINTER, Save R2,R3,R4,R8	: 1069
54	0000V	CF	9E 00002	MOVAB	CHECK_POINTER, R4	
	04	AC	D5 00007	TSTL	COUNT	: 1125
		04	12 0000A	BNEQ	1\$	
			FEFF 0000C	BUGW		: 1126
			0000* 0000E	.WORD	<BUG\$_MAPCNTZER!4>	
		52	D4 00010	CLRL	PLACEMENT	: 1130
53		02	D0 00012	MOVL	#2, POINTER_SIZE	: 1131
04		6C	91 00015	CMPB	(AP), #4	: 1132
		08	1F 00018	BLSSU	2\$	
52	10	AC	D0 0001A	MOVL	PLACEMENT_CODE, PLACEMENT	: 1135
		02	13 0001E	BEQL	2\$	: 1136
		53	D6 00020	INCL	POINTER_SIZE	: 1137
58	0C	AC	D0 00022	MOVL	FILE_HEADER, HEADER	: 1144
00000100	8F	04	AC	D1 00026	CMP	COUNT, #256
		2F	1A 0002E	BGTRU	4\$	: 1146
00400000	8F	08	AC	D1 00030	CMP	LBN, #4194304
		25	1E 00038	BGEQU	4\$	
		53	DD 0003A	PUSHL	POINTER_SIZE	: 1149
64		01	FB 0003C	CALLS	#1, CHECK_POINTER	



			62		50	E9	0003F		BLBC	R0, 7\$		
					52	D5	00042		TSTL	PLACEMENT	:	1150
					03	13	00044		BEQL	3\$	:	
69	02		89		52	B0	00046		MOVW	PLACEMENT, (MAP_POINTER)+	:	1153
	89		0E		01	F0	00049	3\$:	INSV	#1, #14, #2, (MAP_POINTER)	:	1157
		04	AC		01	83	0004E		SUBB3	#1, COUNT, (MAP_POINTER)+	:	1158
89	06		00	0A	AC	F0	00053		INSV	LBN+2, #0, #6, (MAP_POINTER)+	:	1159
			89	08	AC	B0	00059		MOVW	LBN, (MAP_POINTER)+	:	1160
					67	11	0005D		BRB	10\$	:	1146
					53	D6	0005F	4\$:	INCL	POINTER_SIZE	:	1166
		00004000	8F	04	AC	D1	00061		CMPL	COUNT, #16384	:	1167
					28	1A	00069		BGTRU	6\$	:	
					53	DD	0006B		PUSHL	POINTER_SIZE	:	1170
			64		01	FB	0006D		CALLS	#1, CHECK_POINTER	:	
			57		50	E9	00070		BLBC	R0, 11\$	:	
					52	D5	00073		TSTL	PLACEMENT	:	1171
					03	13	00075		BEQL	5\$	:	
			89		52	B0	00077		MOVW	PLACEMENT, (MAP_POINTER)+	:	1174
69	02		0E		02	F0	0007A	5\$:	INSV	#2, #14, #2, (MAP_POINTER)	:	1178
	50		AC		01	C3	0007F		SUBL3	#1, COUNT, R0	:	1179
89	0E	04	00		50	F0	00084		INSV	R0, #0, #14, (MAP_POINTER)+	:	
					AC	D0	00089		MOVL	LBN, 1(MAP_POINTER)	:	1180
		01	A9	08	05	C0	0008E		ADDL2	#5, MAP_POINTER	:	1181
			59		33	11	00091		BRB	10\$	:	1167
					53	D6	00093	6\$:	INCL	POINTER_SIZE	:	1186
		40000000	8F	04	AC	D1	00095		CMPL	COUNT, #1073741824	:	1187
					23	1A	0009D		BGTRU	9\$	:	
					53	DD	0009F		PUSHL	POINTER_SIZE	:	1190
			64		01	FB	000A1		CALLS	#1, CHECK_POINTER	:	
			23		50	E9	000A4	7\$:	BLBC	R0, 11\$	:	
					52	D5	000A7		TSTL	PLACEMENT	:	1191
					03	13	000A9		BEQL	8\$	:	
			89		52	B0	000AB		MOVW	PLACEMENT, (MAP_POINTER)+	:	1194
	50	04	AC		01	C3	000AE	8\$:	SUBL3	#1, COUNT, R0	:	1198
	89		50		10	9C	000B3		ROTL	#16, R0, (MAP_POINTER)+	:	
		FD	A9	C0	8F	88	000B7		BISB2	#192, -3(MAP_POINTER)	:	1199
			89	08	AC	D0	000BC		MOVL	LBN, (MAP_POINTER)+	:	1200
					04	11	000C0		BRB	10\$	:	1187
					FEFF	000C2	9\$:	BUGW			:	1204
					0000*	000C4		.WORD	<BUG\$ PTRCNT!4>		:	
			50		01	D0	000C6	10\$:	MOVL	#1, R0	:	1208
						04	000C9		RET		:	
					50	D4	000CA	11\$:	CLRL	R0	:	1210
					04	000CC		RET			:	

; Routine Size: 205 bytes, Routine Base: \$CODE\$ + 0000



```
223 1211 1 ROUTINE CHECK_POINTER (SIZE) : L_CHECK_POINTER =
224 1212 1
225 1213 1 ++
226 1214 1
227 1215 1 FUNCTIONAL DESCRIPTION:
228 1216 1
229 1217 1 This routine determines whether a map pointer of the given size
230 1218 1 will fit in the file header.
231 1219 1
232 1220 1
233 1221 1 CALLING SEQUENCE:
234 1222 1 CHECK_POINTER (ARG1)
235 1223 1 INPUT PARAMETERS:
236 1224 1 ARG1: map pointer size in words
237 1225 1
238 1226 1 IMPLICIT INPUTS:
239 1227 1 R8: address of file header
240 1228 1
241 1229 1 OUTPUT PARAMETERS:
242 1230 1 NONE
243 1231 1
244 1232 1 IMPLICIT OUTPUTS:
245 1233 1 NONE
246 1234 1
247 1235 1 ROUTINE VALUE:
248 1236 1 1 if pointer fits
249 1237 1 0 if not
250 1238 1
251 1239 1 SIDE EFFECTS:
252 1240 1 header map in use count updated if pointer fits
253 1241 1
254 1242 1 --
255 1243 1
256 1244 2 BEGIN
257 1245 2
258 1246 2 EXTERNAL REGISTER
259 1247 2 HEADER = 8 : REF BBLOCK; ! file header address
260 1248 2
261 1249 2
262 1250 2 IF .HEADER[FH2$B_MAP_INUSE] + .SIZE GTRU
263 1251 2 .HEADER[FH2$B_ACOFFSET] - .HEADER[FH2$B_MPOFFSET]
264 1252 2 THEN RETURN 0;
265 1253 2
266 1254 2 HEADER[FH2$B_MAP_INUSE] = .HEADER[FH2$B_MAP_INUSE] + .SIZE;
267 1255 2
268 1256 2 RETURN 1;
269 1257 2
270 1258 1 END; ! end of routine CHECK_POINTER
```

```
0004 00000 CHECK_POINTER:
51 3A A8 9A 00002 .WORD Save R2
51 04 AC C0 00006 MOVZBL 58(HEADER), R1
ADDL2 SIZE, R1
```

```
: 1211
: 1250
:
```

	50	02	A8	9A	0000A	MOVZBL	2(HEADER), R0	:	1251
	52	01	A8	9A	0000E	MOVZBL	1(HEADER), R2	:	
	50		52	C2	00012	SUBL2	R2, R0	:	
	50		51	D1	00015	CMPL	R1, R0	:	
			09	1A	00018	BGTRU	1\$	:	
3A	A8	04	AC	80	0001A	ADDB2	SIZE, 58(HEADER)	:	1254
	50		01	D0	0001F	MOVL	#1, R0	:	1256
				04	00022	RET		:	
			50	D4	00023	CLRL	R0	:	1258
				04	00025	RET		:	

; Routine Size: 38 bytes,      Routine Base: \$CODE\$ + 00CD

; 271            1259 1  
; 272            1260 1 END  
; 273            1261 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	243	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	30	0	1000	00:01.9

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:MAKPTR/OBJ=OBJ\$:MAKPTR MSRC\$:MAKPTR/UPDATE=(ENH\$:MAKPTR)

; Size:            243 code + 0 data bytes  
; Run Time:        00:11.4  
; Elapsed Time:    00:24.1  
; Lines/CPU Min:   6636  
; Lexemes/CPU-Min: 24226  
; Memory Used:    154 pages  
; Compilation Complete



0171 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

